7. Basic data types

This part explains all the basic data types available in SOFL.

The outline:
- The numeric types
- The character type
- The enumeration type
- The boolean type
- An example of using those types
The numeric types include:

- **nat0** -- \{0, 1, 2, 3, \ldots\}  naturals containing zero.
- **nat**  -- \{1, 2, 3, \ldots\}  naturals
- **int**  -- \{\ldots, -2, -1, 0, 1, 2, \ldots\}  integers
- **real** -- \{\ldots, -2.5, -1.4, 0.0, 1.4, 2.5, \ldots\}  real numbers

The operations on the numeric types are given on the next slide.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>- x</td>
<td>Unary minus</td>
<td>real --&gt; real</td>
</tr>
<tr>
<td>abs(x)</td>
<td>Absolute value</td>
<td>real --&gt; real</td>
</tr>
<tr>
<td>floor(x)</td>
<td>Floor</td>
<td>real --&gt; int</td>
</tr>
<tr>
<td>x + y</td>
<td>Addition</td>
<td>real * real --&gt; real</td>
</tr>
<tr>
<td>x - y</td>
<td>Subtraction</td>
<td>real * real --&gt; real</td>
</tr>
<tr>
<td>x * y</td>
<td>Multiplication</td>
<td>real * real --&gt; real</td>
</tr>
<tr>
<td>x / y</td>
<td>Division</td>
<td>real * real --&gt; real</td>
</tr>
<tr>
<td>x div y</td>
<td>Integer division</td>
<td>int * int --&gt; int</td>
</tr>
<tr>
<td>x rem y</td>
<td>Remainder</td>
<td>int * int --&gt; nat0</td>
</tr>
<tr>
<td>x mod y</td>
<td>Modulus</td>
<td>nat0 * nat0 --&gt; nat0</td>
</tr>
<tr>
<td>x ** y</td>
<td>Power</td>
<td>real * real --&gt; real</td>
</tr>
</tbody>
</table>
Examples: let $x = 9, y = 4.5, z = 3.14, a = -4, b = 3.$
Then
- $-z = -3.14$
- $\text{abs}(a) = 4$
- $\text{floor}(y) = 4$
- $x + z = 12.14$
- $x - y = 4.5$
- $a \times b = -12$
- $x / y = 2.0$
- $a \text{ div } b = -1$
- $a \text{ rem } b = 1$
- $x \text{ mod } b = 0$
- $x \times b = 729$
The relational operators over numeric types are:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Type</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>x &lt; y</td>
<td>Less than</td>
<td>real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x &gt; y</td>
<td>Greater than</td>
<td>real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x &lt;= y</td>
<td>Less or equal</td>
<td>real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x &gt;= y</td>
<td>Greater or equal</td>
<td>real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x &lt; y &lt; z</td>
<td>Less-between</td>
<td>real * real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x &lt;= y &lt;= z</td>
<td>Less-equal-between</td>
<td>real * real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x = y</td>
<td>Equal</td>
<td>real * real</td>
<td>bool</td>
</tr>
<tr>
<td>x &lt;&gt; y</td>
<td>Not equal</td>
<td>real * real</td>
<td>bool</td>
</tr>
</tbody>
</table>
The character type
char

A value of char type: 'x'

Examples:
'a'  'B'  '|'  ')'  ':'  '@'  '7'
All the characters:

English letters:

a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Other characters:

, . : ; * + - / _ ~ | ￥ ( ) [ ] { } @ ^ ` ' & % $ # " ! < > = ?

Newline

White space
The enumeration type

An enumeration type is a finite set of special values, usually with the feature of describing a systematic phenomena.

For example:

Week = {<Monday>, <Tuesday>, <Wednesday>, <Thursday>, <Friday>, <Saturday>, <Sunday>}

There is no operator on the enumeration type.
If we declare a variable `weekday` with the type `Week` as

```plaintext
weekday: Week;
```

then the variable can take any value of the type, that is, `weekday` can take `<Monday>`, `<Tuesday>`, `<Wednesday>`, and so on.
## The boolean type

```plaintext
bool = \{true, false\}
```

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>and</td>
</tr>
<tr>
<td>or</td>
<td>or</td>
</tr>
<tr>
<td>not</td>
<td>not</td>
</tr>
<tr>
<td>=&gt;</td>
<td>implies</td>
</tr>
<tr>
<td>&lt;=&gt;</td>
<td>if and only if</td>
</tr>
</tbody>
</table>

These operators also apply to undefined value `nil`. 
An example of using those types

A simple process telling fares of railway tickets for different kinds of passengers:

```process Tell_Fare(passenger: {<STUDENT>, <ORDINARY>, <PENSIONER>}) fare: real
ext rd normal_fare: real
post fare = case passenger of
  <STUDENT> --> normal_fare - 0.25 * normal_fare;
  <ORDINARY> --> normal_fare;
  <PENSIONER> --> normal_fare - 0.30 * normal_fare
end_case
end_process;```
Exercise 7

1. Let $x = 12$, $y = 9.8$, $z = 2$, and $a = -20$. Evaluate the expressions:
   
   a. $- z$
   b. $\text{abs}(a)$
   c. $\text{floor}(y)$
   d. $x + z$
   e. $x - y$
   f. $a * z$
   g. $x / y$
   h. $a \text{ div } z$
   i. $a \text{ rem } x$
   j. $x \text{ mod } z$
   k. $x ** z$
2. Let $x = 20$, $y = 5.5$, $z = 'd'$, and $a = true$. Evaluate the expressions:
   a. $'a' = z$
   b. $')' <> z$
   c. $x >= y$
   d. $x < y <= y$
   e. $a = false$
   f. $a <> true$

3. Assume that the courses to teach on weekdays are:
   "Software Engineering" on Monday, "Program Design" on Tuesday, "Discrete Mathematics" on Wednesday, "Programming Language" on Thursday, and "Formal Engineering Methods" on Friday. Write a formal specification for the process that gives the corresponding course title for an input weekday.